DNN (deep neural network)

# 1. 회귀모형

neuralnet 패키지를 이용한 BostonHousing 데이터 분석

library(neuralnet)  
require(Metrics)

## Loading required package: Metrics

data("Boston", package = "MASS")  
data = Boston  
  
keeps = c("crim", "indus", "nox", "rm", "age", "dis", "tax",   
 "ptratio", "lstat", "medv")  
data = data[keeps]  
data = scale(data)  
  
apply(data, 2, function(x) sum(is.na(x)))

## crim indus nox rm age dis tax ptratio lstat   
## 0 0 0 0 0 0 0 0 0   
## medv   
## 0

f = medv~ crim + indus + nox + rm + age + dis + tax +   
 ptratio + lstat  
  
set.seed(2016)  
n = nrow(data)  
train = sample(1:n, 400, FALSE)  
  
fit = neuralnet(f, data = data[train, ],   
 hidden = c(10, 12, 20), #히든유닛의 개수가 중요  
 algorithm = "rprop+", # resilent backpropagation with backtracking  
 err.fct = "sse",  
 act.fct = "logistic", #사실 이건 필요가없는데…라고하심  
 threshold = 0.1,   
 linear.output = TRUE)  
  
pred = compute(fit, data[-train, 1:9])  
  
round(cor(pred$net.result, data[-train, 10])^2, 6)

## [,1]  
## [1,] 0.809458

mse(data[-train, 10], pred$net.result) #MSE값~

## [1] 0.2607601849

rmse(data[-train, 10], pred$net.result) #아마..root MSE

## [1] 0.5106468299

deepnet 패키지

require(deepnet)

## Loading required package: deepnet

set.seed(2016)  
X = data[train, 1:9]  
Y = data[train, 10]  
  
fitB = nn.train(x=X, y=Y,  
 initW = NULL, # 가중치에 대한 초기값  
 initB = NULL, # 편의항에 대한 초기값  
 hidden = c(10, 12, 20),  
 learningrate = 0.58,  
 momentum = 0.74,  
 learningrate\_scale = 1, # 학습률과 관련된 파라미터들  
 activationfun = "sigm",  
 output = "linear",  
 numepochs = 970,  
 batchsize = 60, # 배치 크기   
 hidden\_dropout = 0,  
 visible\_dropout = 0)  
  
Xtest = data[-train, 1:9]  
predB = nn.predict(fitB, Xtest)  
  
round(cor(predB, data[-train, 10])^2, 6)

## [,1]  
## [1,] 0.930665

mse(data[-train, 10], predB)

## [1] 0.08525447959

rmse(data[-train, 10], predB)

## [1] 0.2919836975

# 2. 분류모형

mbench의 PimaIndiansDiabetes2 데이터: National Institude of Diabetes and Digestive and Kidney Diseases에서 수집한 데이터로 Pima 인디어 유산(heritage)의 21세이상 여성에 대한 768개의 관측값과 9개의 변수로 구성

library(RSNNS)

## Loading required package: Rcpp

data("PimaIndiansDiabetes2", package="mlbench")  
str(PimaIndiansDiabetes2)

## 'data.frame': 768 obs. of 9 variables:  
## $ pregnant: num 6 1 8 1 0 5 3 10 2 8 ...  
## $ glucose : num 148 85 183 89 137 116 78 115 197 125 ...  
## $ pressure: num 72 66 64 66 40 74 50 NA 70 96 ...  
## $ triceps : num 35 29 NA 23 35 NA 32 NA 45 NA ...  
## $ insulin : num NA NA NA 94 168 NA 88 NA 543 NA ...  
## $ mass : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 NA ...  
## $ pedigree: num 0.627 0.351 0.672 0.167 2.288 ...  
## $ age : num 50 31 32 21 33 30 26 29 53 54 ...  
## $ diabetes: Factor w/ 2 levels "neg","pos": 2 1 2 1 2 1 2 1 2 2 ...

sapply(PimaIndiansDiabetes2, function(x) sum(is.na(x)))

## pregnant glucose pressure triceps insulin mass pedigree age   
## 0 5 35 227 374 11 0 0   
## diabetes   
## 0

temp = PimaIndiansDiabetes2  
temp$insulin = NULL # 결측치 많은 두 변수 및 결측치 제거   
temp$triceps = NULL  
temp = na.omit(temp)  
  
y = (temp$diabetes)  
levels(y) = c("0", "1") #분류변수를 다시 0, 1로 코딩함  
y = as.numeric(as.character(y))  
  
temp$diabetes = NULL  
temp = cbind(temp, y)  
temp = scale(temp)  
  
set.seed(2016)  
n = nrow(temp)  
n\_train = 600  
n\_test = n - n\_train  
train = sample(1:n, n\_train, FALSE)  
  
require(RSNNS)  
  
set.seed(2016)  
X = temp[train, 1:6]  
Y = temp[train, 7]  
  
fitMLP = mlp(x = X, y = Y, size = c(12, 8), maxit = 1000,  
 initFunc = "Randomize\_Weights",   
 initFuncParams = c(-0.3, 0.3),   
 learnFunc = "Std\_Backpropagation",   
 learnFuncParams = c(0.2, 0),   
 updateFunc = "Topological\_Order",  
 updateFuncParams = c(0),  
 hiddenActFunc = "Act\_Logistic",  
 shufflePatterns = TRUE,  
 linOut = TRUE)  
  
predMLP = sign(predict(fitMLP, temp[-train, 1:6]))  
table(predMLP, sign(temp[-train, 7]),   
 dnn = c("Predicted", "Observed"))

## Observed  
## Predicted -1 1  
## -1 67 9  
## 1 21 27

error\_rate = (1 - sum(predMLP == sign(temp[-train, 7]))/124)  
round(error\_rate, 3)

## [1] 0.242

AMORE 패키지

detach("package:RSNNS", unload=TRUE)  
library(AMORE)  
  
net = newff(n.neurons = c(6, 12, 8, 1), # 6: 입력변수, 1: 출력노드  
 learning.rate.global = 0.01,  
 momentum.global = 0.5,  
 error.criterium = "LMLS", # min log squared error (robust)  
 Stao = NA,  
 hidden.layer = "sigmoid",   
 output.layer = "purelin",  
 method = "ADAPTgdwm") # adaptive gradient descent with momentum  
  
X = temp[train, 1:6]  
Y = temp[train, 7]  
  
fit = train(net, P = X, T = Y, error.criterium = "LMLS",  
 report = TRUE, show.step = 100, n.shows = 5)

## index.show: 1 LMLS 0.239138435481238   
## index.show: 2 LMLS 0.236182280077741   
## index.show: 3 LMLS 0.230675203275236   
## index.show: 4 LMLS 0.222697557309232   
## index.show: 5 LMLS 0.214651839732672

pred = sign(sim(fit$net, temp[-train, ]))  
table(pred, sign(temp[-train, 7]),   
 dnn = c("Predicted", " Observed"))

## Observed  
## Predicted -1 1  
## -1 71 10  
## 1 17 26

error\_rate = (1 - sum(pred == sign(temp[-train, 7]))/124)  
round(error\_rate, 3)

## [1] 0.218